Apellidos: **Nombre:** **Matrícula:**

Prueba objetiva 1 - Clave b

Concurrencia

2013-2014 - Primer semestre

Dpto. de Lenguajes, Sistemas Informáticos e Ingeniería de SW
Universidad Politécnica de Madrid

Normas

Este es un cuestionario que consta de **7 preguntas** en **4 páginas**. Todas las preguntas son **preguntas de respuesta simple** excepto la pregunta 7 que es una **pregunta de desarrollo**. La puntuación total del examen es de **10 puntos**. La duración total es de **una hora**. El examen debe contestarse en las **hojas de respuestas**. No olvidéis rellenar **apellidos, nombre y DNI** en cada hoja de respuesta.

Sólo hay una respuesta válida a cada pregunta de respuesta simple. Toda pregunta en que se marque más de una respuesta se considerará incorrectamente contestada. Toda pregunta incorrectamente contestada restará del examen una cantidad de puntos igual a la puntuación de la pregunta dividido por el número de alternativas ofrecidas en la misma.

Cuestionario

(1 punto) 1. Dado el siguiente programa concurrente:

```
static class Hilos {
  static class MiHilo
    extends Thread {
    volatile int n = 0;
    public void run () {
      n++;
    }
  }
```

```
  public static final void
    main(final String[] args) {
    Thread t = new MiHilo();

    t.start();
    t.run();
  }
}
```

Se pide marcar la afirmación correcta.

(a) El máximo número de procesos que pueden ejecutar simultáneamente es tres.

(b) El máximo número de procesos que pueden ejecutar simultáneamente es dos.

(1 punto) 2. Dado el programa concurrente de la pregunta 1.

Se pide marcar la afirmación correcta.

(a) `n++` no es una sección crítica.

(b) `n++` es una sección crítica.

(1 punto) 3. **Se pide** señalar si la siguiente afirmación es cierta o falsa:

El método `signal` de la clase `es.upm.babel.cclib.Semaphore`[1] puede bloquear al proceso que lo invoca.

(a) Falso.

(b) Cierto.

[1] Que es una implementación clásica de semáforos sin nada extraño añadido.

(1 punto) 4. Dado el siguiente programa concurrente:

```
static class Semaforos {
  static Semaphore s =
    new Semaphore(0);

  static class A
    extends Thread {
    public void run () {
      s.await();
      System.io.println(
        "Soy un thread A");
    }
  }
```

```
  static class B
    extends Thread {
    public void run () {
      s.signal();
      System.io.println(
        "Soy un thread B");
    }
  }

  public static final void
    main(final String[] args) {
    Thread a = new A();
    Thread b = new B();

    a.start();b.start();
  }
}
```

Se pide señalar si la siguiente afirmación es cierta o falsa:

El programa podría entregar cualquier de estas dos salidas:

Soy un thread A *Soy un thread B*	*Soy un thread B* *Soy un thread A*

(a) Falso.

(b) Cierto.

(1½ puntos) 5. Se muestra a continuación una implementación de un almacen para un solo dato:

```
class Almacen1 implements Almacen {
  private Producto alm;
  private Semaphore m = new Semaphore(1);
  private Semaphore h = new Semaphore(1);
  private Semaphore d = new Semaphore(0);

  public Almacen1() {
    almacenado = null;
  }
```

```
  public void almacenar(Producto p) {
    m.await();h.await();
    alm = p;
    d.signal(); m.signal();
  }

  public Producto extraer() {
    Producto r;
    m.await();d.await();
    r = alm;
    d.signal(); m.signal();
    return r;
  }
}
```

Se pide señalar al afirmación correcta.

(a) No cumple la propiedad de ausencia de interbloqueo.

(b) Cumple la propiedad de ausencia de interbloqueo.

Apellidos: **Nombre:** **Matrícula:**

(1½ puntos) 6. Supongamos un programa concurrente con cuatro tipos de procesos T_0, T_1, T_2 y T_3 que ejecutan repetidamente operaciones *r.Cero*, *r.Uno*, *r.Dos* y *r.Tres*, respectivamente. Suponiendo que existe al menos un proceso de cada tipo y que *r* es un recurso compartido del tipo especificado a continuación[2]:

C-TAD Circular
TIPO: *Circular* = *(a :* $\mathbb{B} \times$ *b :* $\mathbb{B}$*)*
INVARIANTE: self.$a \vee$ *self*.b

INICIAL: self $=$ (Cierto, Cierto)

CPRE: Cierto
Cero
POST: self $=$ (self^{pre}.a, self^{pre}.b)

CPRE: self.$a \wedge$ self.b
Uno
POST: self $=$ ($\neg$self^{pre}.a, self^{pre}.b)

CPRE: Cierto
Dos
POST: self $=$ (self^{pre}.a, $\neg$self^{pre}.b)

CPRE: $\neg$self.$a \vee \neg$self.b
Tres
POST: self $=$ ($\neg$self^{pre}.a, $\neg$self^{pre}.b)

Se pide señalar la respuesta correcta.

(a) Alguna llamada *r.Uno* podría violar la invariante.

(b) Alguna llamada *r.Dos* podría violar la invariante.

(c) Alguna llamada *r.Tres* podría violar la invariante.

(d) Alguna llamada *r.Cuatro* podría violar la invariante.

[2]No se muestra la declaración de operaciones.

(3 puntos) 7. Se pide especificar un recurso compartido que pretende controlar la carga de un contenedor. El recurso tendrá dos operaciones: *cargar* (indicando un peso) y *retirar* (se retira el contenedor y se repone otro). Inicialmente el peso de la carga en el contenedor es 0. Las operaciones de carga se realizarán mientras no se alcance un peso máximo (*MAXPeso*) almacenado en el contenedor. Cuando se ha superado ese peso máximo ya no se pueden realizar más cargas. La operación de retirar se realizarán cuando se haya superado el peso máximo y volverán a dejar el peso de la carga en el contenedor a 0.

Se pide completar la especificación.

C-TAD Contenedor

OPERACIONES

ACCIÓN cargar: $\mathbb{N}$*[i]*
ACCIÓN retirar:

SEMÁNTICA
DOMINIO:

TIPO: *Contenedor* =

INVARIANTE:

INICIAL:

CPRE:

cargar(p)

POST:

CPRE:

retirar

POST: